Computing Ego Velocity from Scene Flow estimation

Alexis Meyer, Raoul de Charette*

***Abstract*—This technical report describes a new approach that estimates the velocity of the ego vehicle using a monocular setup to compute an approximation of the scene flow.**

***Index Terms*—Computer Vision, Image Flow, Ego motion, Intelligent Transportation System**

I. Introduction

In the last two decades, several methods were proposed to estimate the ego motion of mobile robot as it is helps ego localization, trackings of objects and more generally improves scene understanding. Often ego motion extraction is done using LiDAR [1], [2] sometimes with an optional IMU as in [3], or using vision [4], [5], [6]. For the latter, the fashion way is to use a stereo setup and estimate the displacement of the vehicle from the shift of keypoints in left/right images which provides a stereo disparity and thus RGB and Depth. As highlighted in [7] there is a large number of stereo techniques compared to the few monocular ones though some recent paper were proposed [8], [9], [10].

We investigate in this document a method that requires only a monocular setup and compute the ego motion of the vehicle from the approximation of scene flow using a flat-world assumption. Our approach is novel and - though preliminary - the results obtained are promising. In the following we provide an overview of our methodology, details on our flow computation and the extraction of the optical flow. The document ends with our preliminary evaluations.

II. Methodology

We propose here a new approach to estimate ego velocity using a monocular camera setup. In contradiction to stereo, the task is made significantly harder since monocular vision can not provide any depth information.

To overcome the lack of depth data, we use a strong assumption: the ground is locally planar and all pixels below the horizon line are ground pixels. Using this assumption, we can compute the scene flow (velocity in the scene) from temporal image flow and calibration settings (section III). Although our assumption is strong and fails for many pixels (i.e. any non-ground pixels) we have found that the error induced by the latter is similar to a widely spread noise. Conversely, in the velocity/angle space the pixels validating our assumption (i.e. ground pixels) are well grouped and can be extracted through Gaussian fitting (section IV). Having done this the estimation of the ego motion is straightforward since the ground has a relative speed (i.e. in vehicle reference) equals to the inverse of the world vehicle velocity. We compare also different extraction methods: maximum, and various Mean Shift variations. In section V we evaluate our method on the widely used Kitti datasets [11] and discuss the promising results obtained.

Authors are with the Robotics and Intelligent Transportation Systems (RITS) Team, INRIA Paris, 2 rue Simone Iff, 75012 Paris, France {raoul.de-charette, alexis.meyer} @inria.fr

III. Temporal image flow computation

The aim of this section is to explain how the temporal image flow has been exploited. The word temporal highlights the fact that this method only requires one single camera. The inputs of this method are the camera calibration parameters : internal parameters (focal length, magnification of the objective, optical center position) and the external ones (rotation and translation applied on the camera to pass from the vehicle to the camera coordinate system).

A. 2D image flow

Image flow computes the relative shift of pixel patches from image t to image $t+1$, often through the minimization of a distance function in a restricting neighborhood.

In the literature there is two types of image flow, namely: sparse and dense. The former estimates shift of patches independently and fails for patches that could not be tracked (i.e. non-saliency patches). Dense image flow on the contrary interpolates patch motion to fill missing values. In our study, we used the sparse image flow Farneback method [12] which approximates by a second degree polynomial, according to the following signal model:

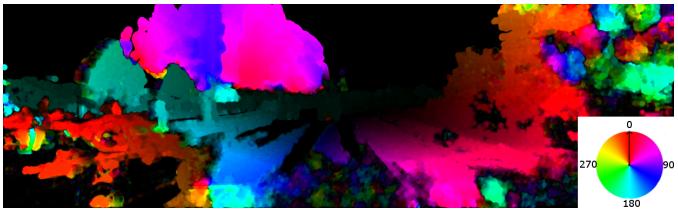
$$f(x) \sim x^T A x + b^T x + c \tag{1}$$

The identification of parameters A, b and c results of a weighted least squares approximation of the signal, where the weighting function normally is a Gaussian. The minimization of the image signal difference at time t and $t+1$ in a given neighborhood provides an estimation of the temporal shift. Figure 1 shows the temporal optical flow on the Kitti database. The following parameters were used: winsize = 9, polygon_n = 5, polygon_sigma = 1.1, iteration_number = 15 (refer to [12] for parameters meaning). Note that to compensate the camera imaging noise a pre-smoothing is applied (9x9 Gaussian kernel).

In figure 1 the angle of the shifting was encoded as Hue whereas shift magnitude is encoded as Brightness. For example, bright red indicates a top right displacement whereas a dark . In the image flow, static elements are eccentric from the optical center (when the car is moving forward) whereas dynamic elements have an almost unpredictable behavior. Conversely, a fast overtaking car will induce a concentric image flow. As expected in such a dynamic scene the optical flow suffers from various problems, mainly: tracking precision

(a) Image at t

(b) Image at $t+1$



(c) Temporal optical flow

Fig. 1. Result of the sparse Farneback image flow [12] on Kitti dataset. Refer to text for details.

due to a lack of saliency, tracking loss due border effects or large image displacement (more likely to occur with low camera frequency). In addition, the projection model implies smaller image shift for far-away displacement rather than close one, thus making harder to detect a dynamic object motion when far away.

The 2D image flow here obtained cannot be exploited as such. In fact, processing a 3D scene flow reveals to be the next step to a better understanding of the scene motion.

B. 3D scene flow

Forward projection geometry tells us that applying the projection matrix to a 3D vector coordinate point (X,Y,Z), we obtain its 2D imaged position (x,y):

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{bmatrix} f_x & s_{uv} & c_u & 0 \\ 0 & f_y & c_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} & R_{3x3} & & t_x \\ & & & t_y \\ & & & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (2)$$

and

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} su \\ sv \\ s \end{pmatrix} \quad (3)$$

where u,v the image coordinates, s a scaling constant, f_x, f_y the focal length in pixels (respectively, width and height), c_u, c_v the coordinates of the optical center in the image, R_{3x3} and t_x, t_y, t_z the 3x3 rotation matrix and translation components to apply for camera to vehicle transformation. Using equation 2 we can compute a forward projection of a 3D point (scene to image), but we require backward projection (image to scene) which in practice can be done due to the loss of depth information in the projection process.

In spite of the unknown depth, we use inverse calibration projection geometry to compute the light ray direction for each pixel. The pseudo-inverse projection matrix M_i is a matrix that if multipled from the right by the projection matrix is equal to the identity. If we multiply from left with M_i, eq. 2 becomes:

$$M_i \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \quad (4)$$

With eq. 4 we are able to obtain the projection vector (X_c,Y_c,Z_c) of the pixel (x,y). Reader will note that because of the loss of s, there is no way to recover the depth along the vector. Instead our assumption is to assume that all pixels below the horizon belong to the ground surface which is approximated as locally planar. That is:

$$a*X+b*Y+c*Z=h \quad (5)$$

With n = (a,b,c) the normal vector of the plane (in our case (0,1,0) because we consider that the flat ground is collinear to the plane Y=0). Hence, the pixel depth can be obtained from the intersection of the projection vector (X_c,Y_c,Z_c) and the ground as expressed in eq. 5. This intersection occurs at (X,Y,Z), with:

$$\begin{cases} X = (X_c - t_x)d + t_x \\ Y = (Y_c - t_y)d + t_y = h \\ Z = (Z_c - t_z)d + t_z \end{cases} \quad (6)$$

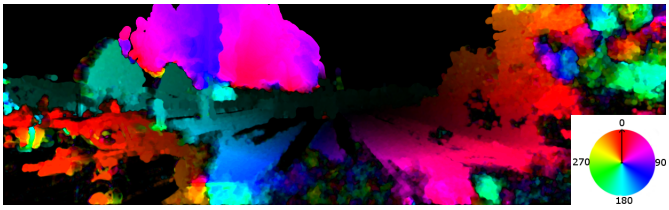
where (X,Y,Z) are the real coordinates of a point of the scene, d the parameter of the system (the unknown variable) and h the altitude of the ground.

Now that we are able to couple one pixel to a position in the scene we can turn the image displacement to scene displacement, in other words compute the scene flow from the image flow. Two components are needed to represent the velocity vector : the magnitude v of the velocity (in km/h) and its orientation angle θ.

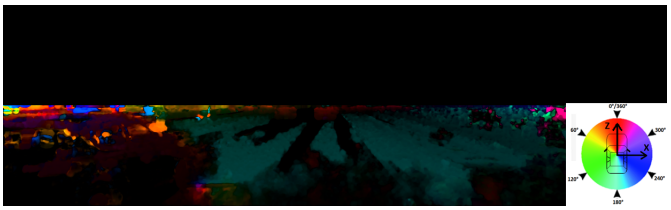
$$\begin{cases} v = |d| * f * 3.6 \\ |\theta| = acos(\frac{V_M . V_n}{|V_M||V_n|}) \\ sign(\theta) = sign(V_{n_X} V_{M_X} - V_{n_Z} V_{M_Z}) \end{cases} \quad (7)$$

where d is the displacement vector of a considered element of the scene (of norm metric the meter) , V_n the unit Z axis vector of the scene and f the frequency of the camera. Note that the multiplication 3.6 if for m/s to km/h conversion.

Figure 2 shows an example output of our scene flow estimation, with the angle (θ) encoded as Hue and velocity (v) encoded as Brightness. In the latter red (that is 0°/360°) correspond to a forward relative motion in the scene, and cyan (that is 180°) is for backward relative scene motion. We can see that despite different optical flow our strong assumption leads to an accurate scene flow computation for all true ground pixels. This is visible in figure 2 where all road



(a) Image flow



(b) Scene flow

Fig. 2. Image flow (top) and our corresponding estimation of the scene flow (bottom). In the latter, the ground is represented with a unified color (dark cyan) which proves that our scene flow estimation is accurate for ground pixels.

pixels have similar dark cyan color (i.e. same scene velocity angle and magnitude). Interestingly reader will notice that even for non ground pixels that can not satisfy our assumption, the estimated scene flow is coherent. For example even if the black overtaking car is not the ground, the orientation angle of the velocity is coherent to the reality.

The 3D scene flow should now be analyzed in order to extract the ego motion of the vehicle.

IV. Ego motion extraction

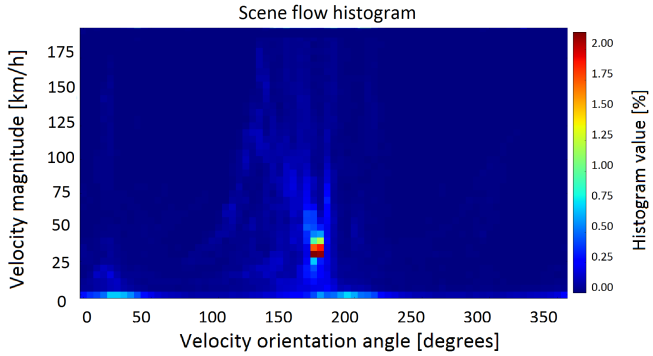


Fig. 3. Scene flow histogram : Discretisation step of 5 km/h and 5 degrees.

Figure 3 shows the histogram of magnitude versus angle of the scene flow velocity. Since many pixels satisfy our assumption the histogram predominant distribution is located at the ground scene velocity location. We use histogram to extract the vehicle ego motion. To extract the velocity from angle/velocity distribution we compare three extraction methods: a) Mean Shift, b) Mean shift with multiple seeds, c) Global maximum.

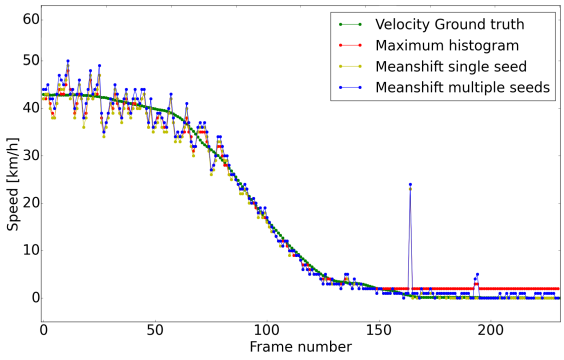
a) *Mean Shift* density estimator [13]. The principle of the latter is to converge iteratively towards the maximum of the distribution, assuming an underlying local Gaussian distribution. As for most density estimator the Mean Shift requires an initialization seed that is critical. In our case we used the maximum histogram as the initial seed. One of the main drawbacks of mean shift is the risk to extract a local maximum.

b) *Mean Shift with multiple seeds* was implemented to leverage the local maximum risk. Our 8 seeds our radially distributed on a circle centered on the maximum of the histogram and with radius equal to the uncertainty limit of the model. Our estimate here is the barycenter of the 8 meanshifts convergence.

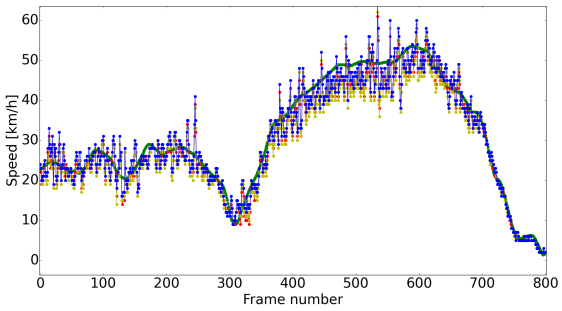
c) *Global Maximum* was also used as a naive comparison.

We now discuss the evaluation of our complete methods and compare the different extraction approaches.

V. Evaluation



(a) Extraction on DB1



(b) Extraction on DB3

Fig. 4. Ego motion extraction on DB1 and DB3 (cf. text for naming).

To evaluate the proposed method we used the Kitti benchmark database with the ad-hoc calibrations settings, and the associated velocity ground truth. Three datasets were evaluated from the training database 2 that are sequence 2, 8 and 9. Respectively refered as DB1, DB2, DB3. The complete extraction results of DB1 and DB3 are displayed in figure 4.

For all three databases we also detail the RMSE (Root Mean Square Error) which is a frequently used metric to measure the differences between two curves: here the ground truth and our extracted ego velocity.

Besides the complexity of the tested datasets (shadows, light saturation, lack of saliency, non-planar surfaces, traffic), the

RMSE (Km/h)	DB1	DB2	DB3	Mean
Maximum histogram	2.42	2.73	3.48	2.87
Mean Shift one seed	2.52	3.47	4.07	3.35
Mean Shift multiple seeds	2.49	2.85	3.41	2.91

Fig. 5. RMSE calculation for each database. The maximum histogram seems to have the best results with in average 2.87 km/h.

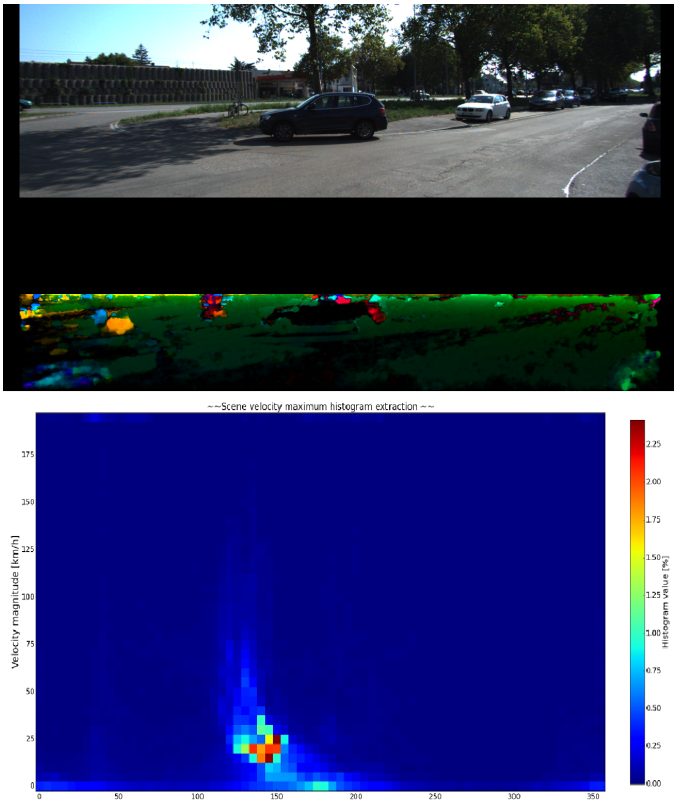


Fig. 6. On the top the car turns to the right, in the middle the corresponding scene flow, in the bottom the scene flow histogram. When the car turns, the velocity vectors correctly calculated through the method turn to the opposite direction.

extracted velocities are coherent to the reality. When the car is turning, the estimated gaussian is curving to the opposite way as in fig. 6, which is logical since we observe the inverse ego motion vectors. Even when the car stops, the extracted ego motion reaches 0 km/h. Those comments underline the validity of our approach. The database DB 3 has 800 frames and many changes during the sequence (motion changes, lateral motion) whereas the two others are linear (no lateral motion, less changes) explaining its higher RMSE (DB 3 column in figure 5). In fact while the car meets changes, the image flow tracking algorithm performances can decrease even slightly causing the scene flow velocities to be wrongly estimated. For high velocity, the pitch angle of the car changes abruptly (due to relief, pot holes, etc.) causing temporal inconsistencies locally in the scene flow extraction. As a consequence the extraction is more noisy for higher velocities.

In term of quantitative results, we were surprised to note that the maximum histogram reveals to be the best extraction among the three implemented. Nevertheless the mean shift multiple obtains close results, but the use of barycenter is probably not sufficient and better clustering should be applied for higher precision.

VI. Conclusions

This paper aimed to reveal an interesting and innovative way of ego motion extraction in the context of autonomous driving. The intentional use of a single camera setup made this approach challenging. Although the results are really preliminary and require additional work, they look promising and validate our strong assumption. Future works should consider using a more complex extraction method and a temporal filtering. Undoubtedly, this will improve performances.

Acknowledgment

This work has been conducted during an internship at INRIA Paris under the supervision of Raoul de Charette.

References

[1] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2174–2181.

[2] J. Zhang, M. Kaess, and S. Singh, "Real-time depth enhanced monocular odometry," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 4973–4980.

[3] S. Scherer, J. Rehder, S. Achar, H. Cover, A. Chambers, S. Nuske, and S. Singh, "River mapping from a flying robot: state estimation, river detection, and obstacle mapping," *Autonomous Robots*, vol. 33, no. 1-2, pp. 189–214, 2012.

[4] I. Cvišić and I. Petrović, "Stereo odometry based on careful feature selection and tracking," in *Mobile Robots (ECMR), 2015 European Conference on*. IEEE, 2015, pp. 1–6.

[5] M. Buczko and V. Willert, "Flow-decoupled normalized reprojection error for visual odometry."

[6] ——, "How to distinguish inliers from outliers in visual odometry for high-speed automotive applications," in *Intelligent Vehicles Symposium (IV), 2016 IEEE*. IEEE, 2016, pp. 478–483.

[7] F. Raudies and H. Neumann, "A review and evaluation of methods estimating ego-motion," *Computer Vision and Image Understanding*, vol. 116, no. 5, pp. 606–633, 2012.

[8] M. H. Mirabdollah and B. Mertsching, "Fast techniques for monocular visual odometry," in *German Conference on Pattern Recognition*. Springer, 2015, pp. 297–307.

[9] S. Song, M. Chandraker, and C. C. Guest, "Parallel, real-time monocular visual odometry," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4698–4705.

[10] S. Song and M. Chandraker, "Robust scale estimation in real-time monocular sfm for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1566–1573.

[11] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2012, pp. 3354–3361. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6248074

[12] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*. Springer, 2003, pp. 363–370.

[13] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," jan 1975.